MAASAI MARA UNIVERSITY

REGULAR
UNIVERSITY EXAMINATIONS

2019/2020 ACADEMIC YEAR

THIRD YEAR SECOND SEMESTER

SCHOOL OF SCIENCE AND INFORMATION SCIENCES
DEPARTMENT OF COMPUTING AND INFORMATION SCIENCE
BACHELOR OF SCIENCE IN INFORMATION SCIENCES
COURSE CODE: COM - 2103
COURSE TITLE: OBJECT ORIENTED PROGRAMMING
DATE:6TH 12 2019 TIME:8:30 – 10:30

INSTRUCTION TO CANDIDATE

i. Question ONE in section A is compulsory
ii. Answer any OTHER Two (2) Questions from section B
iii. Use diagrams, example and illustration where necessary
iv. All questions in section B have equal marks

SECTION A: COMPULSORY 30 MARKS

a) Explain the following terms object **[2 marks]**
b) Real world objects have two parts, state and discuss using appropriate example in C++. **[4 marks]**
c) Why Object Technology? **[4 Marks]**
d) With appropriate example, explain and distinguish Declarations and Definitions in C++
e) With appropriate example in C++ define *inline functions* **[4 Marks]**
f)

Ans: The real world consists of objects. Computer programs may contain computer world representations of the things (objects) that constitute the solutions of real world problems.

Ans: Properties (or state: characteristics that can change),
Behavior (or abilities: things they can do).

g) Why Object Technology?

ANS: Expectations are, Reducing the effort, complexity, and cost of development and maintenance of software systems. Reducing the time to adapt an existing system (quicker reaction to changes in the business environment). Flexibility, reusability. Increasing the reliability of the system.

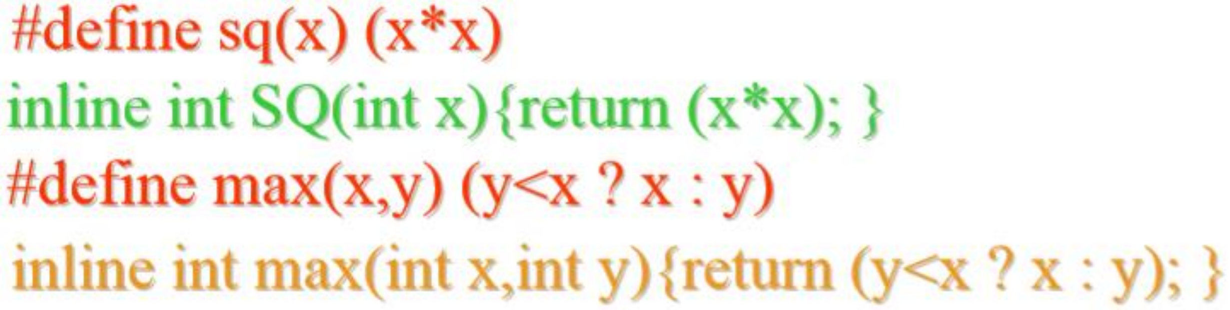
Ans: **A declaration introduces a name –an identifier –to the compiler. It tells the compiler "This function or this variable exists somewhere, and here is what it should look like." ▶A definition, on the other hand, says: "Make this variable here"or "Make this function here."It allocates storage for the name.**

Declarations

```
extern int i;// Declaration
int i;// Definition
struct ComplexT{    // Declaration float re,
im; }; ComplexT c1,c2;    // Definition
void func( int, int);   // Declaration(its body is a definition)
```

In C, declarations and definitions must occur at the beginning of a block. ▶In C++ declarations and definitions can be placed anywhere an executable statement can appear, except that they must appear prior to the point at which they are first used. This improve the readability of the

program. ▶A variable lives only in the block, in which it was defined. This block is the scope of this variable.
Ans: In C++ macros are defined as normal functions. Here the keyword inline is inserted before the declaration of the function.



```
#define sq(x) (x*x)
inline int SQ(int x){return (x*x); }
#define max(x,y) (y<x ? x : y)
inline int max(int x,int y){return (y<x ? x : y); }
```

An inline function is defined using almost the same syntax as an ordinary function. However, instead of placing the function's machine-language code in a separate location, the compiler simply inserts it into the location of the function call. : intj, k, l ; // Three integers are defined ………// Some operations over k and l j = max(k, l); // inline function max is inserted inline inline int intmax max((int intx x, ,int inty y){return ( ){return (y<x ? x : y) y<x ? x : y); } ; } #define #definemax max((x x,y ,y) ( ) (y<x ? x : y y<x ? x : y)) j= j=((k<l ? k : l)

a) Define the term Function Overloading **[2 marks]**

b) Define the term inheritance as used object oriented programming and distinguish between ***base class*** and ***derived class***. **[6 Marks]**

Inheritance is one of the ways in object-oriented programming that makes reusability possible. ▶Reusability means taking an existing class and using it in a new programming situation. ▶By reusing classes, you can reduce the time and effort needed to develop a program, and make software more robust and reliable.
The old class (called the base class base class) is not modified, but the new class (the derived class derived class) can use all the features of the old one and additional features of its own

a) **Provide** inheritance syntax, and demonstrate with appropriate example in C++ **[4 Marks]**

Example: Modeling teachers and the principal (director) in a school. ▶First, assume that we have a class to define teachers, then we can use this class to model the principal. Because the principal is a teacher. class Teacher Teacher{ // Base class private:// means public for derived

class members string name; int age, numberOfStudents; public: void setName (const string & new_name){ name = new_name; } }; class Principal Principal:public Teacher Teacher { // Derived class string schoolName;// Additional members intnumberOfTeachers; public: void setSchool(conststring & s_name){ schoolName= s_name; } };

SECTION B: ATTEMP ANY TWO QUESTIONS [40 MARKS]

a) Define the term Encapsulation and Data Hiding and explain explicitly and implicitly in C++ [4 Marks]

Ans: Data and its functions are said to be encapsulated into a single entity. An object's functions, called member functions in C++ typically provide the only way to access its data. The data is hidden, so it is safe from accidental alteration.

b) Consider the following: A Point on a plane has two properties; x-y coordinates. Abilities (behavior) of a Point are, moving on the plane, appearing on the screen and disappearing. Write a C++ program for A model for 2 dimensional points with the following parts: Two integer variables (x ,y) to represent x and y coordinates A function to move the point: move, A function to print the point on the screen: print, A function to hide the point: hide. **[8 Marks]**

Ans: Point point1, point2, point3; : point1.move(50,30); point1.print();

c) In reference to **question (f)** above, write a C++ program that accepts the results of N subjects and calculate the *sum* and *average*.

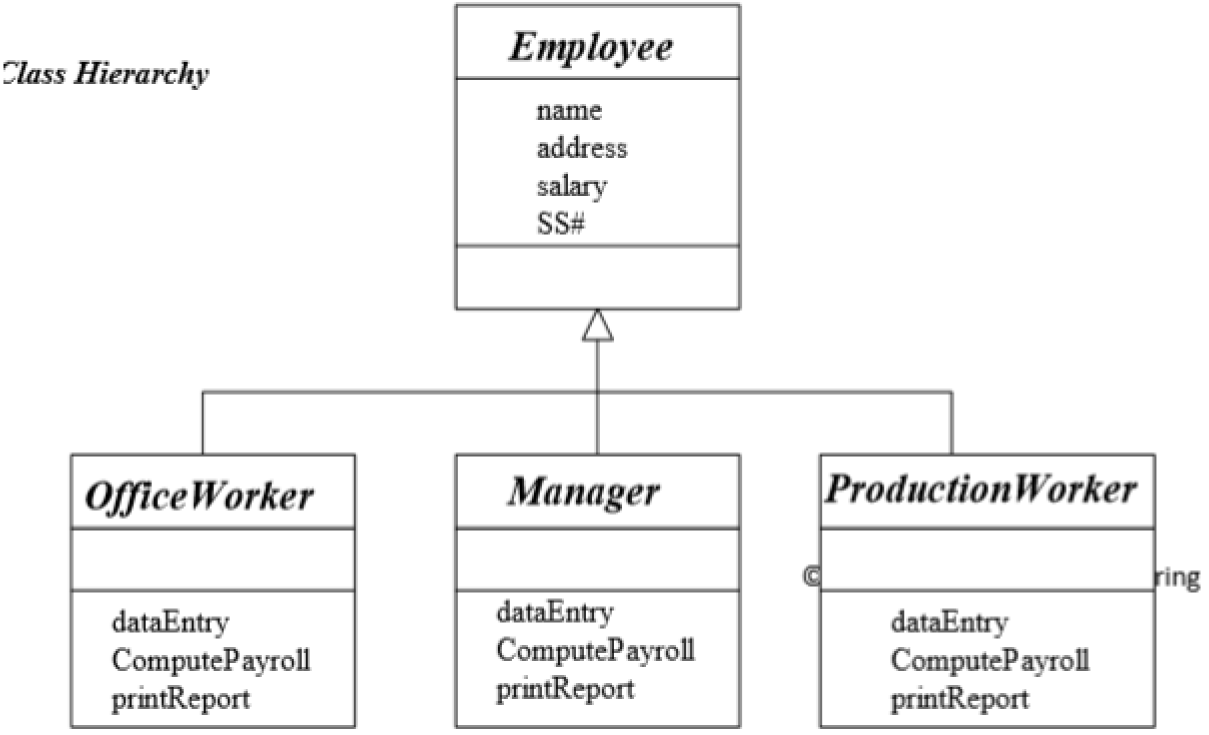
Function Overloading doubleaverage average(constdoublea[],intsize) ; doubleaverage average(constinta[],intsize) ; doubleaverage average(constinta[], constdoubleb[],intsize) ;

doubleaverage(constinta[],intsize) { doublesum= 0.0 ; for(inti=0;i<size;i++)   sum+= a[i] ; return((double)sum/size) ; }

Consider a payroll program that processes employee records at a small manufacturing firm. This company has three types of employees:

 Managers: Receive a regular salary. Office Workers: Receive an hourly wage and are eligible for overtime after 40 hours. Production Workers: Are paid according to a piece rate.

Ans: Some general candidate classes are: • Persons • Places • Things,



QUESTION FOUR [20MARKS]

a) How is a *class* **initialized** in C ++

The class designer can guarantee initialization of every object by providing a special member function called the constructor. ►The constructor is invoked automatically each time an object of that class is created (instantiated). ►These functions are used to (for example) assign initial values to the data members, open files, establish connection to a remote computer etc. ►The constructor can take parameters as needed, but it cannot have a return value (even not void).

a) Distinguish between **Default Constructor** and **Constructors with Parameters** with appropriate demonstration in C++. **[6 Marks]**

A constructor that defaults all its argumentsor requires no arguments, i.e. a constructor that can be invoked with no arguments. class Point{// Declaration Point Class int x,y;// Properties: x and y coordinates public: Point();// Declaration of the default constructor bool move(int, int);// A function to move points void print();// to print coordinates on the screen }; Point::Point(){// Default Constructor cout << "Constructor is called..." << endl; x = 0;// Assigns zero to coordinates y = 0; } intmain(){ Point p1, p2;// Default construct is called 2 times Point *pp = new Point;// Default construct is called once **OR any other appropriate C++ example .**

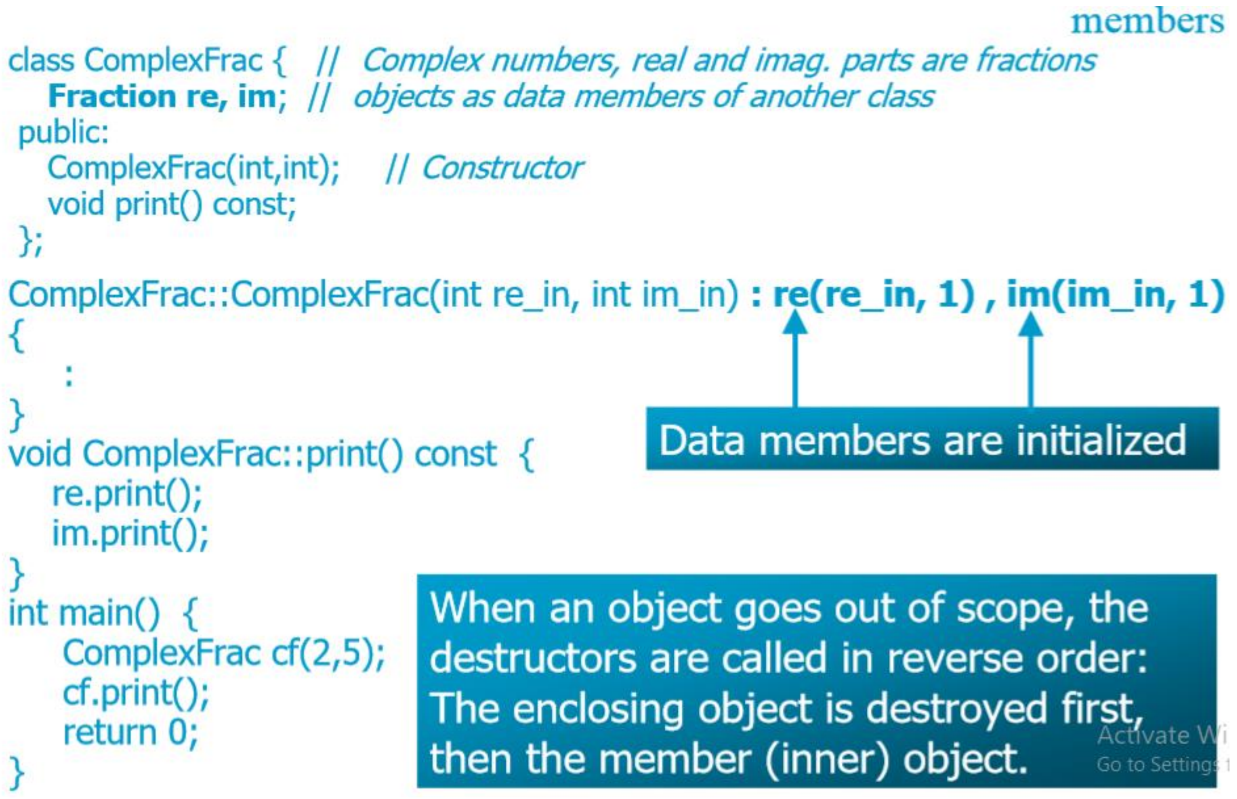
Like other member functions, constructors may also have parameters. ►Users of the class (client programmer) must supply constructors with necessary arguments. class Point{// Declaration Point Class int x,y;// Properties: x and y coordinates public: Point(int, int);// Declaration of the constructor bool move(int, int);// A function to move points void print();// to print coordinates on the screen };

►This declaration shows that the users of the Point class have togive two integer arguments while defining objects of that class. **OR any other appropriate C++ example.**

Point::Point(int x_first, int y_first){ cout << "Constructor is called..." << endl; if (x_first < 0)// If the given value is negative x = 0;// Assigns zero to x else x = x_first; if (y_first < 0)// If the given value is negative y = 0;// Assigns zero to x else y = y_first; } // --------Main Program ------------int main(){ Point p1(20,100), p2(-10,45);// Construct is called 2 times Point *pp = new Point(10,50);// Construct is called once Point p3;// ERROR! There is not a default constructor : }

a) Define the terms *Composition* &*Aggregation* with appropriate example demonstrate using C++. **[11 Marks]**

class Fraction{ // A class to define fractions int numerator,denominator; public: Fraction(int, int);// CONSTRUCTOR void print() const; }; Fraction::Fraction(int num, int denom) {// CONSTRUCTOR numerator=num; if (denom==0) denominator=1; else denominator=denom; cout << "Constructor of Fraction" << endl; } void Fraction::print() const{ cout << numerator << "/" << denominator << endl;}



members

```
class ComplexFrac {  //  Complex numbers, real and imag. parts are fractions
   Fraction re, im;  //  objects as data members of another class
 public:
   ComplexFrac(int,int);    // Constructor
   void print() const;
 };
ComplexFrac::ComplexFrac(int re_in, int im_in) : re(re_in, 1) , im(im_in, 1)
{
    :
}
void ComplexFrac::print() const  {
    re.print();
    im.print();
}
int main()  {
     ComplexFrac cf(2,5);
     cf.print();
     return 0;
}
```

Data members are initialized

When an object goes out of scope, the destructors are called in reverse order: The enclosing object is destroyed first, then the member (inner) object.